# METHOD AND APPARATUS FOR NESTED CONTROL FLOW

## FIELD OF THE INVENTION

**[0001]** The present invention relates generally to arithmetic operations and more specifically to parallel processing on single instruction multiple data stream computations.

## BACKGROUND OF THE INVENTION

**[0002]** In single instruction multiple data (SIMD) parallel processing models, systems are designed to perform the same computation on many sets of data in parallel. Because SIMD processors have impressive cost to performance ratios, they are typically well suited to graphics processing. A typical SIMD processor consists of a single control unit and a set of processing elements where each element is a fully functional arithmetic logic unit capable of executing instructions. The processing element contains local data stored either on local memory or local registers and the control unit determines the instructions for all processing elements. Each processing element, therefore applies an identical computation to a different set of data.

**[0003]** While many graphics problems can be formulated as identical data computations over large sets of data, some computations require different operations and therefore need to support various levels of control flow. Due to the basic operations of SIMD processors, nested control flow may be problematic. A solution for supporting a single level of conditional control flow includes adding a predicate condition, also referred to as a context bit, for each processing element. When the processing element attempts to write a value, it initially checks the context bit and then does not write the element when the context bit is in an off state. A single predicate bit per processing element is an inexpensive implementation in hardware but is limited in only providing a single level of conditional nesting.

**[0004]** Another option to handle nested conditional flow is utilizing a separate control processor to modify the context bit. Although, utilizing a separate control processor is expensive with the requirement of extra processing elements and can slowdown processing speeds. Although, this approach utilizing separate control processors may be utilized in a super computer it is not a feasible solution in a standard processing system.

**[0005]** Another option is utilizing a stack of bits per processing element in lieu of the single context bit. In one approach, a specialized stack per processing element may add significant cost to the device and the stack itself requires additional instructions to manipulate the stack. Among other things, a push command, a pop command and possibly other instructions that modify the stack are internally required.

**[0006]** As the values on the stack correspond to the processing element being on or off, the values on the stack are not independent. Therefore, either the entire stack contains on values or the bottom of the stack contains on values and the top of the stack contains any arbitrary number of off values. Therefore, another approach is to replace each of the stacks (one stack per processing element) by a set of counters (one counter per processing element). The value in each counter would indicate the number of off settings on the stack relative to a transition stage, such as going from an on to an off value. This approach is beneficial as the use of a set of counters requires less hardware. N bit counter can hold the same information as a $2^N$ bit stack. As the amount of hardware required decreases, this approach still has several limitations. Among other things, certain constructs require a compiler to compute additional information, such as to break from a nested loop requires knowing the exact number of control flow constructs that need to be exited by the break. Furthermore, when the amount of hardware has decreased, an additional counter is needed for each processing element. If the element is pipelined, a counter is required

for each pipeline stage in the processing element. Therefore, since many graphics program do not require control flow, this additional hardware adds significant costs without always improving performance.

**[0007]** As such, there exists a need for allowing SIMD parallel processing in a graphics application for performing data computations over large sets of data with nested control flow.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0008]** FIG. 1 illustrates a schematic block diagram of an apparatus for nested control flow in accordance with one embodiment of the present invention;

**[0009]** FIG. 2 illustrates a flow chart of a method for nested control flow in accordance with one embodiment of this present invention;

**[0010]** FIG. 3 illustrates a block diagram of an alternative embodiment of the graphics processing device allowing for a nested control flow in accordance with one embodiment of the present invention; and

**[0011]** FIG. 4 illustrates a flow chart of another method for nested control flow in accordance with another embodiment of the present invention.

## DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

**[0012]** Generally, the present invention includes a method and apparatus for nested control flow including a processor having a context bit. The processor may be, but not limited to, a single processor, plurality of processors, a DSP, a microprocessor, ASIC, state machine, or any other implementation capable of processing and executing software. The term processor should not be construed to refer exclusively to hardware capable of executing software, and may implicitly include DSP hardware, ROM for storing software, RAM, and any other volatile or

non-volatile storage medium. The context bit may be a singular binary bit, such as a zero or one value.

**[0013]** The method and apparatus further includes a first memory device storing a plurality of instructions wherein each of the plurality of instructions includes a plurality of extra bits, the processor operative to execute the plurality of instructions. The memory may be, but not limited to, a single memory, a plurality of memory locations, shared memory, CD, DVD, ROM, RAM, EEPROM, optical storage, microcode or any other non-volatile storage capable of storing digital data for use by the processor. Moreover, the plurality of instructions may be any suitable coded instructions encoded within any suitable programming language or other instructional operation.

**[0014]** The method and apparatus for nested control flow further includes a second memory device operably coupled to the processor, the memory device receiving and incrementing a counter instruction upon the execution of one of the plurality of instructions. The method and apparatus, by the inclusion of the plurality of extra bits in conjunction with the context bit, provide for an improved nested control flow operation through the determination of whether to execute an instruction based not only on the plurality of extra bits, but also the value of the context bit itself.

**[0015]** More specifically, FIG. 1 illustrates a processing device 100 having a processor 102, a first memory device 104 and a second memory device 106. In one embodiment, the processing device 100 may be a shader disposed within a pixel processing pipeline in a graphics processing system. The processor 102 includes a context bit 108 which is stored within a context bit memory location associated with the processor 102. Although, as recognized by one having ordinary skill in the art, the context bit 108 may be stored in any suitable location for access by

the processor 102. Furthermore, second memory device 106 may be a non-dedicated memory device for use with the processor 102, such as but not limited to, a general purpose register.

**[0016]** In accordance with one embodiment of the present invention, the first memory device 104 provides an instruction 110 to the processor 102. In response to the instruction 110, the processor 102 seeks to implement a single instruction operation on multiple data streams. Not illustrated in FIG. 1, the processor 102 further receives the multiple data streams for the performance of the single operation 110. In accordance with one embodiment of the present invention, the instruction 110 includes a plurality of extra bits. In the preferred embodiment, the extra bits include two extra bits, but as recognized by one having ordinary skill in the art, any suitable number of extra bits may be utilized to provide for the below-described functionality.

**[0017]** Upon receipt of the instruction 110, the processor 102 performs the initial operation of determining whether to read the context bit 108 based on the extra bits within the instruction 110. In one embodiment, the first bit is a conditional yes or no bit and the second bit may be a true or false context check bit. Therefore, based on the extra bits within the instruction, if it is determined to examine the context bit 108, the context bit is read. The context bit 108, in one embodiment, indicates either an on or off position. If the context bit 108 is on, the processor 102 therefore executes the instruction on the data set and provides a maintaining counter instruction 112 to a counter (not illustrated) stored within the second memory device 106. In one embodiment, the counter stored within the second memory device 106 may be a simple integer based counter. The counter provides for the number of operations for determining a nested control flow upon receiving another instruction 110, wherein the counter value indicates a nesting depth of context bits that are set to a second state. The processor 102 maintains the

counter within the second memory device 106. Therefore, the counter allows for tracking the number of executed operations.

**[0018]** When the processor 102 has executed all the instructions 110 or is determined to exit a nested control flow, the processor 102 thereupon generates an output signal 114 which is provided in one embodiment to the next step in the pixel processing pipeline. In one embodiment, the processing device 100 may be a shader such that the output is a plurality of shaded vertices.

**[0019]** FIG. 2 illustrates one embodiment of a flow chart of a method for nested control flow. The method begins, step 120, by setting a context bit to either a first state or a second state, step 122. In one embodiment, the first state may be an on state and the second state may be an off state. Step 124 is receiving a first instruction having a plurality of extra bits. As discussed above, with respect to FIG. 1, the first instruction 110 may be received from the memory device 104, wherein the instruction 110 includes the plurality of extra bits.

**[0020]** Step 126 is determining whether to read the context bit based on the plurality of extra bits. This step is performed, in one embodiment, by the processor 102 based on examining the status of the extra bits, such as a true or false state or a yes or no state. Furthermore, if the context bit 108 is read, the context bit is extracted from the memory location storing the context bits 108 such as the processor 102 may read the context bit 108. Step 128 is executing the instruction when the context bit is in the first state, upon the reading of the context bit. As discussed above, the instruction is performed by the processor 102 which is a SIMD processor processing the single instruction upon multiple data sets. Thereupon, the method is complete, step 130.

**[0021]** FIG. 3 illustrates a further graphical representation of the apparatus for nested control flow including a control unit 150, a first ALU 152, a second ALU 154 and a third ALU 156, wherein each ALU includes a context bit 158, 160 and 162 respectively. Furthermore, the ALUs 152, 154 and 156 are coupled to general purpose registers 164, 166 and 168 respectively. As recognized by one having ordinary skill in the art, the general purpose registers 164, 166 and 168 may be any suitable non-dedicated memory which is accessible by the ALUs 152, 154 and 156. The ALUs 152, 154 and 156 receive a single instruction with extra bits 110 from the control unit 150. The first ALU 152 receives a first data set 170, the second ALU 154 receives a second data set 172 and the third ALU 156 receives a third data set 174.

**[0022]** FIG. 3 illustrates three representative ALUs 152, 154 and 156, as recognized by one having ordinary skill in the art in a SIMD processing system, any suitable number of ALUs may be implemented to be operably coupled to receive the single instruction with extra bits 110 and further coupled to general purpose registers, such as 164, 166 and 168, for the storage of computation information therein. Moreover, further ALUs within the system would include the context bit, such as 158, 160 or 162. It should be noted that the three ALUs are for illustrative purposes and not to be meant as so limiting herein.

**[0023]** The present invention utilizes a single context bit per processing element 152, 154 and 156. With the addition of two added bits to each instruction, an instruction can execute independent of the context bit 158, 160 and 162 or can check the context bit 158, 160 and 162 execute only when the context bit 158, 160 and 162 is set to execute, such as an on position. The present invention eliminates the need for a per element counter by using a general purpose register to hold the counter, such as illustrated as the second memory device 106 of FIG. 1. Consequently, any ALU 152, 154 or 156 operations can be used to modify the counter. The

general purpose register used for the counter can be non-dedicated and the same register does not need to be utilized for different kinds of execution of an overall program.

**[0024]** Although, in one embodiment to the present invention, five additional instructions are added to support common conditional flow sequences as noted in the following table:

| Instruction | Action |
|---|---|
| Rout = Push cond, Rin | If cond and Rin == 0 then<br>Rout = 0 |
| Rout = Invert Rin | If Rin == 1 then<br>Rout = 0 |
| Rout = Pop Rin | Rout = Rin –1<br>If Rout <= 0 then |
| Rout = Clear | Rout = large number<br>Set Pred bit to skip |

**[0025]** As recognized by one having ordinary skill in the art, these control flow sequences can be implemented using any combination of instruction from standard arithmetic operations. Using an exemplary program to illustrate operation of the present invention, an example if then statement is the operation of:

If x is greater than 0 then

Y = 3

Y = u+v

**[0026]** With respect to FIG. 3, representative storage locations within the general purpose registers 164, 166 and 168 have been designated as memory locations 180-184, 186-190 and 192-196 respectively. Based on the operation of the ALU 152, 154 and 156, specific data is written within the general purpose register memory locations 184, 186-190 and 192-196. A first operation is the register performing a predicate_push operation to determine if a value stored at

Rx is greater than zero, such as looking at an initial value stored within a register for x performing a predicate_push for register location 180, 186 and 192. Based on this comparison, a second register value may be computed as three (p) relative to register locations 181, 187 and 193. Thereupon, the computation of y=u+v may be performed by the defining of the register value Ry as being the equivalent of the register value Ru in summation with the register value Rv (p). Therefore, register value 180, 186 and 192 would then be defined as a predicate_pop for the value within register 180, 186 and 192.

**[0027]** As recognized by one having ordinary skill in the art, the above example indicates one single nested operation within a control flow, wherein the present invention is utilized within multiple nested control flow operations. Therefore, in a nested control flow operation with multiple nested operations, the counter is implemented such that the number of nested operations into the depth of nested control flow may be effectively monitored and controlled when executing any nested flow operation. More specifically, allow for a machine level instruction set for breaking out of an instruction and jumping around.

**[0028]** As discussed above, there exists any suitable implementation of conditional statements, although two common statements are an if then statement or a while statement. Included below in Table 2 are two representative examples of the implementation of operations wherein the sequences can be nested and other comment control flow sequences can be implemented as well.

| Source sequence | Implementation |
|---|---|
| If cond then<br>Statement 1<br>Else<br>Statement 2<br>Endif | Rstk = Pred_push cond, Rstk<br>Statement1 (p)<br>Rstk = Prev_stack_invert Rstk<br>Statement2 (p)<br>Rstk – Pred_pop Rstk |

| | |
|---|---|
| While (cond)<br>Statement 1<br>Break<br>Statement 2<br>Endwhile | Rsave = Rstk<br>loop<br>Rstk = Push cond, Rstk<br>Statement 1 (p)<br>Rstk = clear<br>Statement 2 (p)<br>Endloop<br>Rstk = restore Rsave |

**[0029]** FIG. 4 illustrates a flow chart of another embodiment of a method for nested control flow. The method begins 200, in one embodiment with the step of resetting a counter value step 202. In one embodiment, the counter value may be stored in a general purpose register for indicating an increment of nested control flow operations. Step 204 is setting a context bit to either a first state or second state. The context bit is associated with the particular arithmetic logic unit, such as ALU 152 of FIG. 3 and is set to either an on or off state or may be set to an execute or off state.

**[0030]** Step 206 is receiving a first instruction having a plurality of extra bits. As discussed above, the single instruction with extra bits 110 may be provided from a control unit 150 to all of the ALUs within a SIMD processor. The extra bits allow for a determination of whether the context bit should be read. Step 208 is determining whether to read the context bit based on the plurality of extra bits. This may be operated in accordance with the operation described above. If the determination is yes, step 210 is executing instructions when the context bit is in a first state, such as an execute state. The single instructions with extra bits 110 are executed in parallel by the various ALUs on various data sets, 170, 172 and 174. Step 212 is maintaining the counter value wherein the counter value indicates a nesting depth of context bits that are set to a second state in a general purpose register to indicate that a calculation has been performed.

**[0031]** In the event that the determination of step 208 is in the negative, the method is complete, step 214. The method is also complete upon the performance of step 212.

**[0032]** As such, the present invention provides for an improved nested control flow through the operation of an instruction set having extra bits for the specialized instructions. Moreover, the present invention utilizes general purpose registers and does not require any excess memory locations within the processors. As such, through utilizing special instructions, the present invention improves over the prior art through utilizing less memory resources and allows for nested control flow and jumping around various loops of instructions in a SIMD environment.

**[0033]** It should be understood that the implementation of other variations and modifications of the invention in its various aspects will be apparent to those of ordinary skill in the art, and that the invention is not limited by the specific embodiments described herein. For example, the general purpose register may be any suitable non-dedicated memory device operative to provide data storage and communication with the arithmetic logic units. It is therefore contemplated and covered by the present invention any and all modifications, variations or equivalents that fall within the spirit and scope of the basic underlying principles disclosed and claimed herein.